Please amend the claims as follows:

CLAIMS

(CURRENTLY AMENDED) 1. A method of saving an interlocking trees data store from memory to permanent storage comprising the steps of:

traversing the interlocking trees data store to access each node;

creating a node packet containing all information relevant to the node;; and

writing the node packet to permanent storage;

wherein, the interlocking trees datastore structure comprising elemental root nodes, subcomponent nodes and end-product nodes using asCase and asResult bi-directional links for determining paths within the interlocking trees datastore structure.

(ORIGINAL) 2. The method of claim 1 wherein said saving of an interlocking trees data store from memory to permanent storage further comprises the step of:

saving supporting structures to permanent storage.

(ORIGINAL) 3. The method of claim 2 wherein the step of saving supporting structures comprises saving any of the following list of structures needed to restore the interlocking trees data store to memory, wherein said list includes:

KStore name, creation date, version/cycle of Save program that created the save file, ~~OS~~ operating system underlying structure information including at least size of fields used information, sign structure information if not saved below, elemental root nodes ~~or~~ and elemental root node values and pointers to the elemental root nodes, levels and associated delimiters;

meta data including one or more of the following field types:

user defined types, column descriptions, and permissions;;

kState variables including one or more of the following:

switches, data streams, and sign structure information for instance special ordering for asCase lists;

data sources including one or more of the following:

types, locations, affiliated data streams)– for learning new knowledge security including one or more of the following:

administrator passwords, user passwords, permissions, saved query locations, and triggers~~,~~ ; and

XML-related meta data~~, if any~~.

(ORIGINAL) 4. ~~A~~ The method of saving an interlocking trees data store from memory to permanent storage according to claim 2, wherein saving supporting structures comprises the steps of:

determining which informational structures will be saved with the interlocking trees data store, and,

formatting and writing said informational structures to permanent storage.

(ORIGINAL) 5. ~~A~~ The method of saving an interlocking trees data store from memory to permanent storage according to claim 1, wherein creating a node packet containing all information relevant to the node, comprises the steps of:

storing the node's current load address in the packet;

storing the Case and Result pointers, any other additional fields, the asCase list of pointers and the asResult list of pointers in the packet; and

writing the node packet to permanent storage.

(ORIGINAL) 6. The method of claim 5 wherein prior to storing any packets, memory is allocated for each packet to be stored.

(ORIGINAL) 7. ~~A~~ The method of saving an interlocking trees data store from memory to permanent storage according to claim 1, wherein traversing the interlocking trees data store to access each node comprises the steps of:

traversing the interlocking trees data store to access each node starting from the primary root, using a typical tree traversal along the asCase paths.

(CURRENTLY AMENDED) 8. A method of saving an interlocking trees data store from memory to permanent storage according to claim 1, wherein traversing the interlocking trees data store to access each node comprises the steps of:

traversing the interlocking trees data store to access each node beginning from an end product nodes~~,~~ .

(ORIGINAL) 9. The method of claim 8 wherein said traversing beginning from end product nodes begins after obtaining access to all end product nodes from a file of end product node information associated with said interlocking trees datastore.

(ORIGINAL) 10. ~~A~~ The method of saving an interlocking trees data store from memory to permanent storage according to claim 1, wherein traversing the interlocking trees data store to access each node comprises the steps of:

traversing the interlocking trees data store to access each node from a root nodes.

(ORIGINAL) 11. The method of claim 10 wherein said traversing beginning from said root nodes begins after obtaining access to all root nodes from a file of root node information associated with said interlocking trees datastore.

(CURRENTLY AMENDED) 12. A method of restoring an interlocking trees data store from permanent storage to memory comprising the steps of:

allocating memory and reading supporting structures required before the interlocking trees data store is restored, from permanent storage into memory;

reading each node packet and allocating memory for nodes;

creating a translation table of old memory addresses & new memory addresses for each node;

reading each node packet and reconstructing nodes and pointer lists; and

allocating memory and reading supporting structures that require address translation using the translation table to be restored, from permanent storage into memory;

wherein, the interlocking trees datastore comprising elemental root nodes, subcomponent nodes and end-product nodes using asCase and asResult bi-

directional links for determining paths within the interlocking trees datastore structure.

(ORIGINAL) 13. The method of claim 12 wherein said allocating memory and reading support structures step finds elemental root node packets and data from said elemental root nodes on a first pass, and then the remaining steps of claim 12 are executed.

(CURRENTLY AMENDED) 14. A set of instructions executable on a computing system which when executed configure said system to provide the facility to save and restore a trees based datastore, said set of instructions comprising:

a save set having;

a first set to traverse the interlocking trees data store to access each node to be saved;

a second set to create a node packet containing all information relevant to the node to be saved; and

a third set to write the node to be saved as a packet created by the second set to permanent storage connected to said computing system;

wherein, the interlocking trees datastore structure comprising elemental root nodes, subcomponent nodes and end-product nodes using asCase and asResult bi-directional links for determining paths within the interlocking trees datastore structure.

(ORIGINAL) 15. A set of instructions executable on a computing system which when executed configure said system to provide the facility to save and restore a trees based datastore, said set of instructions comprising:

a restore set, having;

instructions to reconstruct metadata; and

an address translation table maintenance and using set for establishing an address translation table to convert addresses between addresses in saved packets and addresses in a restored interlocking trees datastore;

wherein, the interlocking trees datastore structure comprising elemental root nodes, subcomponent nodes and end-product nodes using asCase and asResult bi-directional links for determining paths within the interlocking trees datastore structure.

(ORIGINAL) 16. The set of instructions set forth in claim 15 further comprising:
a save set having:

a first set to traverse the interlocking trees data store to access each node;

a second set to create a node packet containing all information relevant to the node; and

a third set to write the node created by the second set to permanent storage connected to said computing system.

(ORIGINAL) 17. A computer system for running an interlocking trees datastore program so that an interlocking trees data store can function in a main memory of said computer system, said computer system having a program for saving said interlocking trees datastore and a program for restoring said interlocking trees datastore wherein addresses of said interlocking trees datastore and said restored interlocking trees datastore are not the same, said program for restoring said interlocking trees datastore having means to establish an address translation table to translate addresses found in node packets created by said save program to new addresses in said restored interlocking trees datastore.

(CURRENTLY AMENDED) 18. A computer system having an interlocking trees datastore in a memory of said computer system and having a saving means for saving said interlocking trees datastore for later restoration, said saving means comprising:

means for locating and saving all relevant header information including metadata relevant to restoring said interlocking trees data store,

means for locating each node in said interlocking trees data store and

means for saving all data about each located node in a packet form;

wherein, the interlocking trees datastore comprising elemental root nodes, subcomponent nodes and end-product nodes using asCase and asResult bi-directional links for determining paths within the interlocking trees datastore structure.

(ORIGINAL) 19. The computer system of claim 18 wherein said means for saving all data determines ~~discovers~~ a saved size for said packet form of said all data about each located node.

(ORIGINAL) 20. The computer system of claim 19 wherein a total size of a saved interlocking trees datastore saved by said saving means is a function of said saved size for each said packet.

(ORIGINAL) 21. The computer system of claim 18 wherein said each packet contains pointer data pointing to addresses of other nodes of said interlocking trees data store that had been linked to the node from which said each packet is constructed in said means for saving.